Scalable Inference for Hybrid Bayesian Networks with Full Density Estimations

Wei Sun, Kuo-Chu Chang, and Kathryn B. Laskey
The Sensor Fusion Lab, Center of Excellence in C4I
Department of Systems Engineering and Operations Research
George Mason University
Fairfax, VA 22030, U.S.A.
wsun, kchang, klaskey@gmu.edu

Abstract − *The simplest hybrid Bayesian network is Conditional Linear Gaussian (CLG). It is a hybrid model for which exact inference can be performed by the Junction Tree (JT) algorithm. However, the traditional JT only provides the exact first two moments for hidden continuous variables* [1]*. In general, the complexity of exact inference algorithms is exponential in the size of the largest clique of the strongly triangulated graph that is usually the one including all of discrete parent nodes for a connected continuous component in the model. Furthermore, for the general nonlinear non-Gaussian hybrid model, it is well-known that no exact inference is possible. This paper introduces a new inference approach by unifying message passing between different types of variables. This algorithm is able to provide an exact solution for polytree CLG, and approximate solution by loopy propagation for general hybrid models. To overcome the exponential complexity, we use Gaussian mixture reduction methods to approximate the original density and make the algorithm scalable. This new algorithm provides not only the first two moments, but full density estimates. Empirically, approximation errors due to reduced Gaussian mixtures and loopy propagation are relatively small, especially for nodes that are far away from the discrete parent nodes. Numerical experiments show encouraging results.*

Keywords: Hybrid Bayesian networks, message passing, Gaussian mixture

1 Introduction

A Bayesian network (BN) [2] [3] [4] [5] represents a joint probability distribution over a set of random variables (RVs) as a directed acyclic graph (DAG) and a collection of conditional probability distributions (CPD). The nodes of the DAG represent RVs, the arcs represent dependencies, and the CPDs parameterize the distributions. Over the past few decades, BNs have become increasingly popular as a powerful tool for representing uncertainty for complex problems. Although both exact and approximate inference for BN are NP-hard in general [6] [7], a number of inference algorithms have been reported in the literature [8]. However, inference for hybrid models with both discrete and continuous variables has many difficulties and open issues. The simplest hybrid Bayesian network model is called Conditional Linear Gaussian (CLG) [9], in which a discrete node can have continuous children, but a continuous node is not allowed to have discrete child. Given its discrete parents, a continuous variable in CLG has a linear functional relationship with its continuous parents. The Junction Tree algorithm [10] [9] can be applied for CLG to provide a solution with the exact first two moments of posterior distributions for hidden continuous variables, and exact posterior probabilities for hidden discrete variables. It is also possible to obtain the full marginal densities using an extended JT scheme [1]. But in general, the complexity of the Junction Tree algorithm is exponential in the size of the largest clique of the strongly triangulated graph, which is usually the clique determined by the size of state space of all discrete parent nodes in a connected CLG [11]. For a general hybrid BN with nonlinear and/or non-Gaussian RVs, as commonly known, there is no existing method that could produce exact posterior distributions. Hence, one has to rely on approximate methods in that case.

Because of the heterogeneity of variables and arbitrary functional relationships in general hybrid models, approximate methods usually produce errors due to functional transformation, distribution approximation, discretization, or structure simplification. For example, Likelihood Weighting (LW) [12] [13] is a general-purpose simulation algorithm that is model-independent, but has difficulty handling unlikely evidence; The Mixture of Truncated Exponential (MTE) algorithm [14] produces closed form solutions, but in-

[1] With extra computations, [1] presented a scheme of JT that can provide full marginal densities.

troduces errors due to approximating arbitrary distributions using mixtures of truncated exponentials; Hybrid Loopy Propagation algorithm [15] uses mixture of Gaussian to represent continuous messages, and then computes messages by numerical integrations.

In this paper, we are particularly interested in the message passing framework because of its simplicity of implementation and good empirical performance. In [16] [17], we presented a hybrid loopy propagation method that uses network partitioning, and then integrates information via the interface nodes. This hybrid message passing algorithm also allows us to apply any suitable algorithm to an individual sub-network. An advantage of the hybrid algorithm is that it is easier to accommodate an efficient algorithm for inference within homogeneous sub-networks. However, a disadvantage is that we have to conduct inference conditioning on all the discrete parent nodes (i.e., interface nodes). Therefore, the algorithm has an exponential complexity proportional to the product of sizes of discrete parent nodes, which is the same as the Junction Tree algorithm.

Pearl's message passing algorithm [3] is the first exact inference method originally proposed only for polytree discrete Bayesian networks. For networks with loops, applying Pearl's message passing in a "loopy" fashion usually converges and provides good approximate results [18]. For pure continuous networks, similarly, Pearl's algorithm is applicable with continuous message represented in appropriate forms such as the first two moments of Gaussian. However, it lacks an efficient way to exchange messages between discrete and continuous variables in a hybrid network. In this paper, we attempt to develop a unified message passing framework for general hybrid networks without network partitioning or graph transformation. Based on the traditional Pearl's message passing mechanism, we derive formulae for computing messages being exchanged between discrete and continuous variables directly. In the framework, each node in the networks propagates messages to its neighbors and depending on the node type, messages are computed accordingly under various circumstances. We call this new distributed approach Direct Message Passing for Hybrid Bayesian Network (DMP-HBN). A similar approach has been taken in [19] using Gaussian mixtures to approximate hybrid conditional densities. In our method, we use the original CPDs without approximation and provide an exact solution for polytree networks.

The remainder of this paper is organized as follows. Section 2 describes the DMP-HBN algorithm in detail. The scalability of our new algorithm is also addressed in the same section. Section 3 shows our numerical simulation examples. Finally, we summarize and discuss future research in Section 4.

2 Direct Message Passing

This section describes DMP-HBN algorithm in detail. We first briefly review Pearl's original message passing algorithm. We then extend it for general hybrid models.

2.1 Pearl's Message Passing Algorithm

Recall that in a polytree network, any node X d–separates evidence into $\{\mathbf{e}^+, \mathbf{e}^-\}$, where $\mathbf{e}^+$ and $\mathbf{e}^-$ are evidence from the sub-network "above" X and "below" X respectively. Every node in the network maintains two values called λ and π. The λ value of X is the likelihood, defined as:

$$\lambda(X) = P(\mathbf{e}_X^- | X) \tag{1}$$

The π value of X, defined as:

$$\pi(X) = P(X|\ \mathbf{e}_X^+) \tag{2}$$

is the conditional probability distribution of X given $\mathbf{e}_X^+$. It is easy to see that the belief of a node X given all evidence is just the normalized product of its λ and π values:

$$\begin{aligned} BEL(X) &= P(X|\mathbf{e}) = P(X|\mathbf{e}_X^+, \mathbf{e}_X^-) \\ &= \frac{P(\mathbf{e}_X^-|X, \mathbf{e}_X^+)P(X|\mathbf{e}_X^+)P(\mathbf{e}_X^+)}{P(\mathbf{e}_X^+, \mathbf{e}_X^-)} \\ &= \alpha P(\mathbf{e}_X^-|X)P(X|\mathbf{e}_X^+) \\ &= \alpha\lambda(X)\pi(X) \end{aligned} \tag{3}$$

where α is a normalizing constant. In message passing, every node sends λ messages to each of its parents and π messages to each of its children. Based on its received messages, every node updates its λ and π values correspondingly. The general message propagation equations of Pearl's algorithm are the following [3]:

$$\pi(X) = \sum_{\mathbf{T}} P(X|\ \mathbf{T}) \prod_{i=1}^{m} \pi_X(T_i) \tag{4}$$

$$\lambda(X) = \prod_{j=1}^{n} \lambda_{Y_j}(X) \tag{5}$$

$$\pi_{Y_j}(X) = \alpha \left[\prod_{k \neq j} \lambda_{Y_k}(X) \right] \pi(X) \tag{6}$$

$$\lambda_X(T_i) = \sum_X \lambda(X) \sum_{T_k:\ k \neq i} P(X|\mathbf{T}) \prod_{k \neq i} \pi_X(T_k) \tag{7}$$

where $T = (T_1, T_2, ..., T_n)$ are the parents of node X; $Y = (Y_1, Y_2, ..., Y_m)$ are children of node X; $\lambda_{Y_j}(X)$ is the λ message node X receives from its child Y_j,

$\lambda_X(T_i)$ is the λ message X sends to its parent T_i; $\pi_X(T_i)$ is the π message node X receives from its parent T_i, $\pi_{Y_j}(X)$ is the π message X sends to its child Y_j; and α is a normalizing constant.

Equations (4) to (7) are recursive equations, so we need to initialize messages properly to start the message propagation. Again, Pearl's algorithm is originally designed for discrete polytree networks, so these propagation equations are for computing discrete probabilities. When Pearl's algorithm is applied to a pure discrete polytree network, the messages propagated are exact and so are the beliefs of all nodes after receiving all messages. For pure continuous networks with arbitrary distributions, we proposed a method called Unscented Message Passing [20] using a similar framework with different message representations and a new corresponding computation method. However, with both discrete and continuous variables in the model, passing messages directly between different types of variables requires additional techniques.

2.2 Direct Message Passing between Discrete and Continuous Variables

In CLG, a continuous node is not allowed to have any discrete child. Therefore, the only case we need to consider when exchanging message between different types of variables is the case of a continuous node with discrete parents. Without loss of generality, suppose that we have a typical hybrid CPD involving a continuous node X with a discrete parent node D and a continuous parent node U, as shown in Figure 1. Messages sent between these nodes are: (1) π message from D to X, denoted as $\pi_X(D)$; (2) π message from U to X, denoted as $\pi_X(U)$; (3) λ message from X to D, denoted as $\lambda_X(D)$; and (4) λ message from X to U, denoted as $\lambda_X(U)$. In addition, each node needs to maintain its λ and π values.

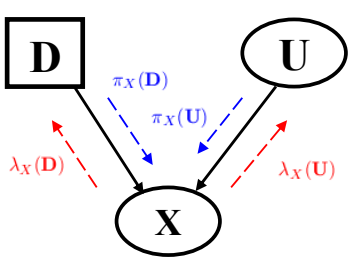


Figure 1: A typical node with hybrid CPD — continuous node X has discrete parent D and continuous parent U.

Let us look at these messages one by one, and derive their corresponding formula based on Pearl's traditional message passing mechanism. First, recall from Equation (6), $\pi_X(D)$ can be computed by substitution:

$$\pi_X(D) = \alpha \left[\prod_{child \neq X} \lambda_{child}(D) \right] \pi(D) \tag{8}$$

where $\lambda_{child}(D)$ is λ message sent to D from each of its children except X, and $\pi(D)$ is the easily computed message sent from the discrete sub-network "above" D. Note that $\lambda_{child}(D)$ is always in the form of a discrete vector. After normalizing, $\pi_X(D)$ is a discrete probability distribution serving as the mixing prior for a Gaussian mixture.

Similarly, but in a different form, $\pi_X(U)$ can be computed as:

$$\pi_X(U) = \alpha \left[\prod_{child \neq X} \lambda_{child}(U) \right] \pi(U) \tag{9}$$

where $\lambda_{child}(U)$ are λ messages sent to U from its continuous children other than X. These λ messages are continuous messages in the form of Gaussian mixtures. $\pi(U)$ is π value of U, and its computation depends on the type of parent nodes it has. The generalized computation of $\pi(X)$ will be described in the next paragraph. Finally, the resulting $\pi_X(U)$ is a normalized product of Gaussian mixtures, resulting in another Gaussian mixture with a greater number of components.

Now for $\pi(X)$, by applying Equation (4) with integral replacing summation for continuous variable, we have,

$$\begin{aligned} \pi(X) &= \sum_D \int_U P(X|D,U)\pi_X(D)\pi_X(U)dU \\ &= \sum_D \left[\pi_X(D) \int_U P(X|D,U)\pi_X(U)dU \right] \end{aligned} \tag{10}$$

where $\pi_X(D)$ and $\pi_X(U)$ are π messages sent from D and U respectively. For a given $D = d$, $P(X|D = d, U)$ defines a probabilistic functional relationship between X and its continuous parent U. The integral of $P(X|D = d, U)\pi_X(U)$ over U is equivalent to a functional transformation of $\pi_X(U)$, which is a continuous message in the form of a Gaussian mixture. In this functional transformation process, we pass each Gaussian component individually to form a new Gaussian mixture. Essentially, $\pi(X)$ is a mixture of continuous distributions weighted by $\pi_X(D)$. To avoid the potential for growing complexity of the message, it is possible to approximate the mixture with a single Gaussian density or a Gaussian mixture with fewer components.

$\lambda(X)$ is relatively straightforward to compute as it is the product of λ messages from each of its children,

which must be continuous variables due to the CLG model restriction. However, since we represent a continuous message as a Gaussian mixture, the product of a set of Gaussian mixtures will be another Gaussian mixture with increased number of components.

Let us now turn to the computation of messages sent from X to its parents D and U. As shown in Equation (7), λ message sent to its parents is essentially an inverse functional transformation of the product of the λ value of the node itself and the π messages sent from all of its other parents via the function defined in the CPD of X. It can be derived as,

$$\lambda_X(D=d) = \int_X \lambda(X) \int_U P(X|D=d,U)\pi_X(U)dUdX \tag{11}$$

where $\int_U P(X|D = d,U)\pi_X(U)dU$ is a functional transformation of a distribution over U into a distribution over X. Further, multiplying by $\lambda(X)$ and integrating over X, results in a non-negative constant, serving as a likelihood of X given $D = d$.

Similarly, the λ message sent from X to its continuous parent U can be expressed as:

$$\begin{aligned}\lambda_X(U) &= \int_X \lambda(X) \sum_D P(X|D,U)\pi_X(D)\pi_X(D)dX \\ &= \sum_D \left[\pi_X(D) \int_X \lambda(X)P(X|D,U)dX\right] \quad (12)\end{aligned}$$

Note that $\int_X \lambda(X)P(X|D,U)dX$ is an integral of the product of X's λ value and its conditional probability distribution; this integral is over X itself. Therefore it results in a density estimate of its parent multiplied by a coefficient. This coefficient is very critical in computing mixing priors with $\pi_X(D)$ when there is more than one component in the mixture distributions.

Equations (8) to (12) form a baseline for computing messages between discrete and continuous variables. Along with the well-defined formulae for computing messages between the same types of variables, they together provide an unified message passing framework for hybrid Bayesian network models. When the network is a polytree, messages propagated between nodes are exact and so the beliefs. When there are loops in the network, DMP-BN still works in the same way as so-called loopy propagation but provides approximate solution. Note that the presence of discrete parents for continuous variable makes the corresponding continuous messages necessarily a mixture distribution. Unfortunately, the number of mixture components in the message increases exponentially with the size of joint state space of the discrete parents. In order to scale the algorithm, one alternative is to combine or reduce the mixture components into smaller ones, trading off complexity against accuracy.

2.3 Complexity and Scalability

The complexity of exact inference for a hybrid model is essentially determined by the size of the joint state space of all discrete parent nodes (i.e., interface nodes). It is easy to prove that, in a connected CLG, all discrete parents will end up in one clique with at least one continuous node [11]. Sometimes, even a CLG with very simple structure can give rise to an intractable clique tree. For example, the network shown in Figure 2 will have all of its discrete nodes in one clique, hence making the computations exponential in the size of the joint state space of all discrete nodes. If each discrete node has 10 states, then the resulting clique will have size 10^n, where n is the number of discrete nodes.

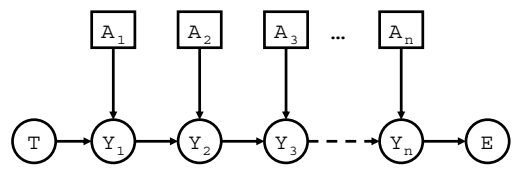


Figure 2: A simple CLG that has an exponential clique tree: $A_1, A_2, ...A_n$ are discrete nodes, and $T, Y_1, Y_2, ..., Y_n, E$ are continuous nodes.

DMP-HBN has the same problem when exact inference is required. This is because for each state of a discrete parent node, its continuous child has to compute messages according to the function defined in the CPD. Therefore, messages sent by a continuous node with a hybrid CPD will be in the form of a Gaussian mixture in which the components are weighted by probabilities passed from its discrete parents. In particular, as shown in Equaion (10) and (12), $\pi(X)$ and $\lambda_X(U)$ are mixtures of Gaussians with the number of Gaussian components equal to the size of the state space of its discrete parent D. When a mixture message propagates to another continuous node with discrete parents, the message size will increase again exponentially. However, while JT has to deal with this intractability, DMP-HBN has the choice to approximate the original Gaussian mixture with a smaller number of components. In many cases, a Gaussian mixture with significantly fewer components can approximate the original density very well. Let us assume that $f(x)$ is the true density, and $\hat{f}(x)$ is the approximate Gaussian mixture. We use the following distance measure as the metric, called Normalized Integrated Square Error (NISE):

$$d = \frac{\int (f(x) - \hat{f}(x))^2 dx}{\int (f(x))^2 dx + \int (\hat{f}(x))^2 dx} \quad .$$

An example shown in Figure 3 demonstrates a reasonable estimate using only 4 components to approximate

a Gaussian mixture with 20 components ($\sqrt{d} < 3\%$). With a pre-defined error bound, Gaussian mixture reduction methods such as the ones proposed in [21, 22] can be applied to find a good approximate mixture with a smaller number of components. It is straightforward to incorporate these methods into DMP-HBN to make the algorithm scalable with an acceptable accuracy trade-off. However, it is non-trivial to estimate the inference error after the messages are compressed and propagated. In the next section, we will provide some performance results with numerical experiments to evaluate the algorithm under various situations.

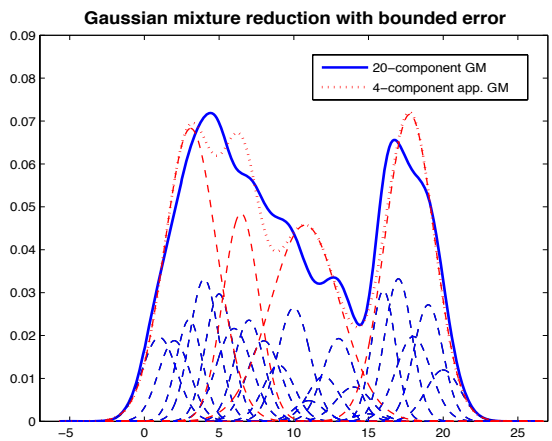


Figure 3: Using a 4-component GM to approximate a 20-component GM with $\sqrt{d} < 3\%$.

3 Numerical Experiments

Theoretically, DMP-HBN can provide exact results for a polytree CLG. For verification purpose, an example model called *Poly12CLG* as shown in Figure 4, was used for the experiment.

Assume evidence is observed on leaf nodes E, and Z. With random observations, we conducted more than 30 independent experiments and compared DMP results with the ones obtained by the Junction Tree algorithm (we used the well-known commercial software Hugin as the JT implementation). The latter algorithm is considered to be the gold standard and the resulting solutions serve as the ground truth. All experiments show that DMP-HBN provides results identical to the ground truth. Furthermore, our algorithm provides full density estimates whereas JT only provides the first two moments for hidden continuous variables. In case of a multi-modal true posterior distribution, full density estimation is obviously preferred. Figure 5 illustrates a true bi-modal posterior distribution example obtained by DMP-HBN, compared to a single Gaussian obtained by JT with the same first two moments.

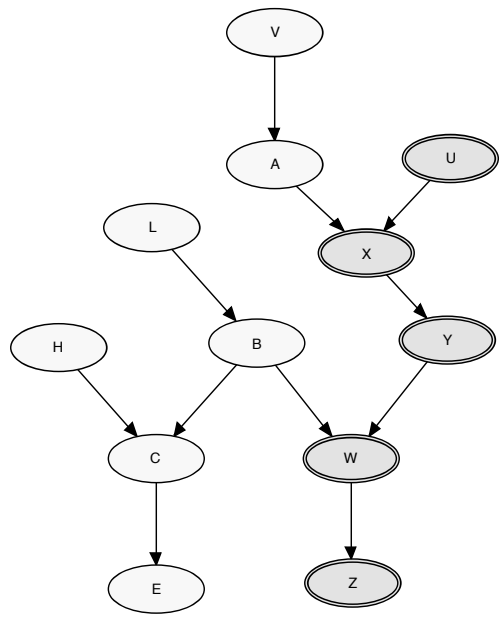


Figure 4: An example polytree CLG model called Poly12CLG, consisting of 7 discrete nodes V, A, L, B, H, C, E and 5 continuous nodes U, X, Y, W, Z.

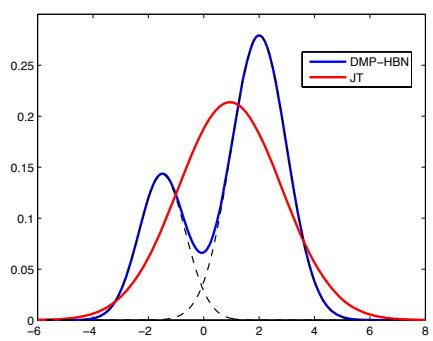


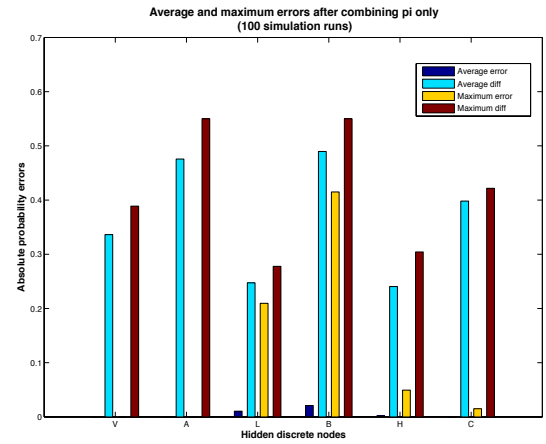
Figure 5: Performance comparison: a bi-modal true posterior density by DMP-HBN, v.s., a single Gaussian estimate by JT.

We also conducted scalability tests of DMP algorithm using the same example model *Poly12CLG*. For many decision support applications, the variables of interest tend to be discrete, such as feature identification, entity classifications, or situation hypotheses. In our experiments, we first show how the assessments of hidden discrete nodes in a CLG are affected after collapsing the Gaussian mixture into a single Gaussian when passing messages. We use average absolute probability errors between two discrete distributions as the metric to evaluate the performance. In general, when a node of interest is relatively far away from the evidence, its posterior distribution would not

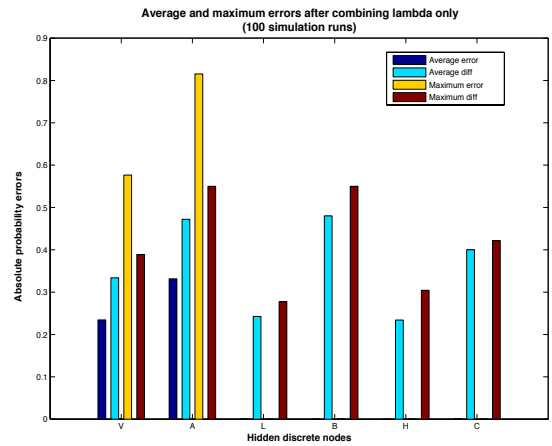
deviate much from its prior. In that case, it is difficult to show the impact of the approximation on the inference error. So we purposely designed CPDs in *Poly12CLG* to move the true posterior probabilites away from its prior. Figure 6 shows the average and maximum errors of the approximate posterior probabilities for hidden discrete nodes V, A, L, B, H, and C, obtained after collapsing Gaussian mixtures into a single term over 100 Monte Carlo simulations. Average and maximum difference between the true posteriors and the priors over these 100 simulations are also shown in the figure for comparison. Figure 6(a) presents the estimate errors when collapsing π values only; Figure 6(b) shows the performance when collapsing λ messages only; and Figure 6(c) displays the inference errors when collapsing both π values and λ messages whenever a mixture of Gaussians is present.

Notice that reducing the π value of a node does not affect the network "above" it because the π message is being sent downward in the network. Similarly, since a λ message is being sent upward, reducing a λ message will not affect the network "below" the node. For example in Figure 6(a), the posterior probabilities of V and A are exact, and in Figure 6(b), the estimates of L, B, H, and C are also exact without inference error. When reducing both π values and λ messages, all posterior distributions are not exact any more. Results shown in Figure 6(c) suggest that the approximation errors diminish when the nodes are farther away from discrete parents. For example, the approximate errors for nodes L, H, and C are very small. However, discrete parent nodes such as A, and B, are affected significantly. This is not surprising due to the relatively large approximation errors when collapsing a multi-modal Gaussian mixture into a single term. One way to achieve a desired accuracy is to specify a pre-defined error bound whenever we try to reduce a Gaussian mixture into one with fewer components. Although it is difficult to perform theoretical analysis of the total inference error after propagation, it is possible to obtain bounded error if the threshold used is small enough. Figure 7 demonstrates significantly better performance for the same model but with the normalized ISE of the reduced Gaussian mixture limited to less than 5% each time. As can be seen from the figure, the average and maximum errors for all nodes are well less than 1%.

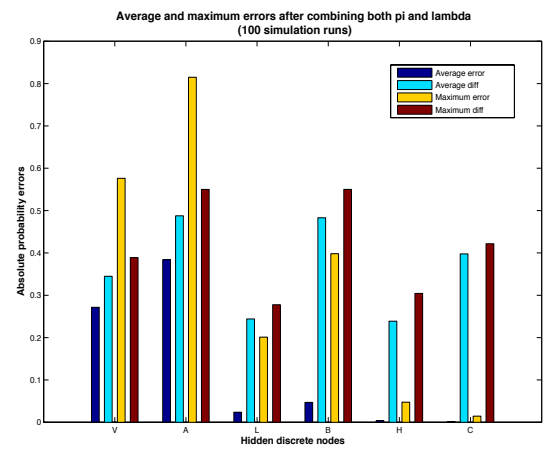
Another example model called *Loop13CLG* (extended from *Poly12CLG*), shown in Figure 8, was used for numerical experimentation on a network with loops. Again, we assume that leaf nodes E and Z are observable evidence nodes. With random observations, Figure 9 shows the average and maximum absolute errors of posterior probabilities for hidden discrete nodes over 100 Monte Carlo simulations. All simulation runs converge in about 11 iterations. As can be seen from the figure, average approximation errors caused by loopy propagation range from less than 1% to about 5% for hidden discrete nodes.



(a) combining π values only



(b) combining λ messages only



(c) combining both π values and λ messages

Figure 6: Scalability test – performance loss after combining Gaussian mixture into one single Gaussian.

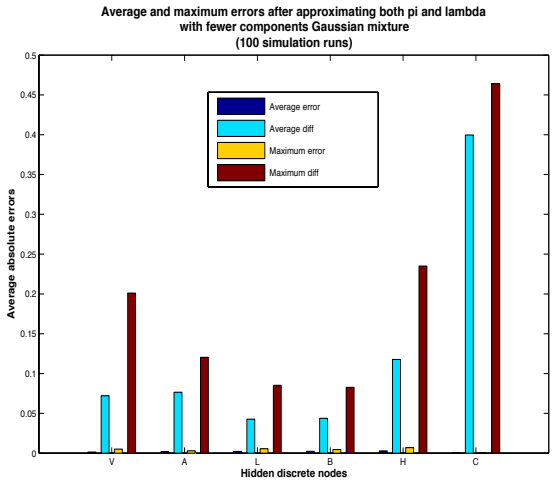


Figure 7: Accurate estimates of the posterior probabilities resulted by limiting approximation error ($<$ 5%) each time when reducing message with fewer components Gaussian mixture.

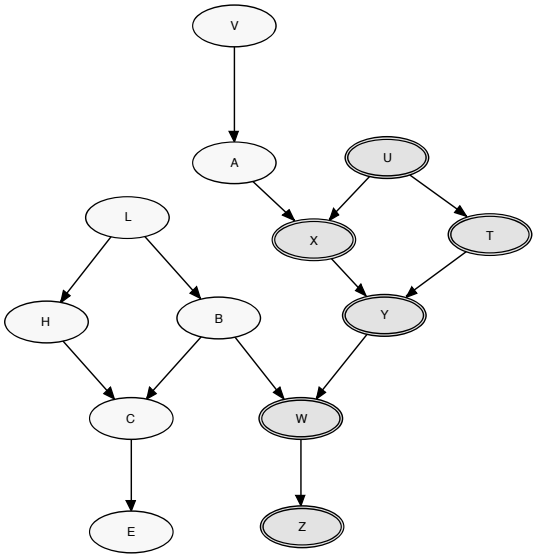


Figure 8: Loop13CLG – an example CLG model with multiple loops, consisting of 7 discrete nodes V, A, L, B, H, C, E and 6 continuous nodes U, X, TY, W, Z.

We also tested DMP with some other networks with randomly pre-defined CPDs. All simulation results suggest that the estimation errors reduce significantly as the node is farther away from the discrete parent nodes.

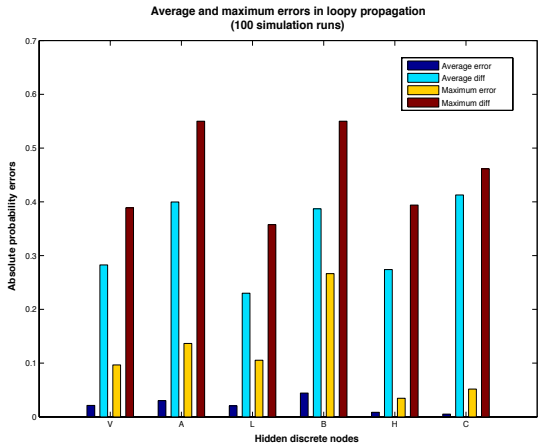


Figure 9: Performance test with loopy CLG model.

4 Summary

In this paper, we present a new representation and inference algorithm called DMP-HBN to exchange messages between discrete and continuous variables. Essentially, this new algorithm provides an alternative for probabilistic inference in hybrid Bayesian networks. It provides full density estimates for continuous variables and can be extended with unscented transformation [20] for the general hybrid models with nonlinear and/or non-Gaussian distributions. Since DMP-HBN is a distributed algorithm utilizing only local information, there is no need to transform the network structure as required by the Junction Tree algorithm. Also recall that our previous works in [16], [17], need to partition the hybrid model into different network segments, each containing only pure type of variables and then conduct message passing separately. Instead, DMP-BN can exchange messages directly between discrete and continuous variables within an unified framework. In addition, the algorithm does not require prior knowledge of the global network topology which could be changing dynamically. This is a major advantage of the algorithm and is particularly important to ensure scalable and reliable message exchanges in a large information network where computations are done locally.

As shown in the empirical simulation results, DMP-HBN is scalable with a performance tradeoff of losing some accuracy. For many decision support applications, we are mainly interested in hidden discrete variables such as entity classifications or high level situation hypotheses. The experimental results show that the estimation errors of the hidden discrete variables depend on the network topology and are relatively modest in the cases considered. Theoretically, it is non-trivial to estimate the performance bounds quantitatively due to message compressing and propagation. This points to an important topic for future

research.

References

[1] S. L. Lauritzen and F. Jensen, "Stable local computations with conditional gaussian distributions," *Statistics and Computing*, vol. 11, no. 2, pp. 191–203, April 2001.

[2] E. Charniak, "Bayesian networks without tears: making bayesian networks more accessible to the probabilistically unsophisticated," *AI Magazine*, vol. 12, no. 4, pp. 50–63, 1991.

[3] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kauffman, San Mateo, 1988.

[4] F. Jensen, *An Introduction to Bayesian Networks*. Springer-Verlag, New York, 1996.

[5] R. Neapolitan, *Probabilistic Reasoning in Expert Systems*. John Wiley & Sons, New York, 1990.

[6] G. Cooper, "The computational complexity of probabilistic inference using bayesian belief networks," *Artificial Intelligence*, vol. 42, pp. 393–405, 1990.

[7] P. Dagum and M. Luby, "Approximating probabilistic inference in bayesian belief networks is np–hard," *Artificial Intelligence*, vol. 60, pp. 141–153, 1993.

[8] H. Guo and W. Hsu, "A survey of algorithms for real-time bayesian network inference," in *AAAI/KDD/UAI–2002 Joint Workshop on Real-Time Decision Support and Diagnosis Systems*, Edmonton, Alberta, Canada, 2002.

[9] S. Lauritzen, "Propagation of probabilities, means amd variances in mixed graphical association models," *JASA*, vol. 87, no. 420, pp. 1098–1108, 1992.

[10] S. Lauritzen and D. Spiegelhalter, "Local computations with probabilities on graphical structures and their applications to expert systems," in *Proceedings of the Royal Statistical Society, Series B.*, vol. 50, 1988, pp. 157–224.

[11] U. N. Lerner, *Hybrid Bayesian Networks for Reasoning about Complex Systems*. Stanford University, October 2002.

[12] R. Fung and K. C. Chang, "Weighting and integrating evidence for stochastic simulation in bayesian networks," in *Uncertainty in Artificial Intelligence 5*. New York: Elsevier Science Publishing Company, Inc., 1989, pp. 209–219.

[13] R. D. Shachter and M. A. Peot, "Simulation approaches to general probabilistic inference on belief networks," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, vol. 5, 1999.

[14] B. R. Cobb and P. P. Shenoy, "Inference in hybrid bayesian networks with mixtures of truncated exponentials," *International Journal of Approximate Reasoning*, vol. 41, pp. 257–286, April 2006.

[15] C. Yuan and M. J. Druzdzel, "Hybrid loopy belief propagation," in *Proceedings of the third European Workshop on Probabilistic Graphical Models*, 2006, pp. 317–324.

[16] W. Sun and K. C. Chang, "Hybrid message passing for general mixed bayesian networks," in *Proceedings of the 10th International Conference on Information Fusion*, Quebec, Canada, 2007.

[17] W. Sun and K. Chang, "Message passing for hybrid bayesian networks: Representation, propagation, and integration," *IEEE Transaction on Aerospace and Electronic Systems*, vol. 45, pp. 1525–1537, Octomber 2009.

[18] K. Murphy, Y. Weiss, and M. Jordan, "Loopy belief propagation for approximate inference:an empirical study," in *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, 1999.

[19] O. C. Schrempf and U. D. Hanebeck, "A new approach for hybrid bayesian networks using full densities," in *Proceedings of the 6th International Workshop on Computer Science and Information Technologies (CSIT 2004)*, vol. 1, Budapest, Hungary, Oct. 2004, pp. 32–37.

[20] W. Sun and K. Chang, "Unscented message passing for arbitrary continuous bayesian networks," in *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, Vancouver, Canada, July 2007.

[21] H. C. Kuo-Chu Chang and C. Smith, "Constraint optimized weight adaptation for gaussian mixture reduction," in *Proceeding of SPIE Conference on Defense, Security, and Sensing*, Orlando,FL, 2010.

[22] O. C. Schrempf, O. Feiermann, and U. D. Hanebeck, "Optimal mixture approximation of the product of mixtures," in *Proceedings of the 8th International Conference on Information Fusion (Fusion 2005)*, vol. 1, Philadelphia, Pennsylvania, 2005, pp. 85–92.